The University of Amsterdam at the TREC 2011 Session Track

Bouke Huurnink Richard Berendsen Katja Hofmann
Edgar Meij Maarten de Rijke

ISLA, University of Amsterdam
http://ilps.science.uva.nl/

Abstract: We describe the participation of the University of Amsterdam's ILPS group in the Session track at TREC 2011.

1 Introduction

The stream of interactions created by a user engaging with a search system contains a wealth of information. For retrieval purposes, previous interactions can help inform us about a user's current information need. Building on this intuition, our contribution to this TREC year's session track focuses on session modeling and learning to rank using session information. In this paper, we present and compare three complementary strategies that we designed for improving retrieval for a current query using previous queries and clicked results: probabilistic session modeling, semantic query modeling, and implicit feedback.

The rest of this paper is structured as follows: we detail our approach in Section 2, followed by our retrieval setup in Section 3. We describe our results in Section 4 and end with a concluding section.

2 Modeling Sessions and Queries, and Learning from Feedback

Here we describe our three strategies for session-based search: probabilistic session modeling, semantic query modeling, and implicit feedback. Before we start, we fix our notation and terminology. We use q_m to denote the current query and $I = \{i_1, i_2, \ldots, i_{m-1}\}$ to denote the set of past interactions, where each interaction i_n is associated with a query q_n, a set of displayed result snippets R_n and a set of clicked result snippets C_n.

2.1 Probabilistic Session Modeling

Our probabilistic session modeling strategy uses interpolation to combine results from the current query with results from previously issued queries and with results from previously displayed or clicked snippets. Our experiments use progressively increasing amounts of session information: first previously issued queries, then previously displayed result snippets, and then previously clicked result snippets. They are aimed at answering the following research questions:

- Does incorporating results from previously issued queries improve performance for the current query?
- Does incorporating results from previously displayed result snippets improve performance for the current query?
- Does incorporating results from previously clicked result snippets improve performance for the current query?

To form a query from a set of displayed or clicked result snippets, we apply RM-1 [6].

Our session modeling approach is based on the FixInt approach of Shen et al. [10]. However, instead of creating new query models, we model the sessions at the document level. We calculate $P(q_m|d)$, the probability of q_m given a document d, and $P(i_n|d)$, the probability of each prior interaction given d. We calculate $P(\mathcal{N}|d)$, the probability of the current information need $\mathcal{N}$, by combining these probabilities using an interpolation parameter α such that:

$$P(\mathcal{N}|d) = \alpha \cdot P(q_m|d) + (1-\alpha) \cdot \frac{\sum_{n=1}^{m-1} P(i_n|d)}{|I|}. \quad (1)$$

As α increases, previous interactions are given less weight. When $\alpha = 1$, only the results from the current query are taken into account. All previous interactions are given equal weight, normalizing by the number of previous interactions.

Further, we model an interaction as a combination of a past query q_n issued by the user, and a feedback query f_n formed from either R_n or C_n, depending on the experimental condition. We interpolate these results again, using a second parameter β:

$$P(i_n|d) = \beta \cdot P(q_n|d) + (1-\beta) \cdot P(f_n|d). \quad (2)$$

Substituting Eq. 2 into Eq. 1, we obtain

$$P(\mathcal{N}|d) = \alpha \cdot P(q_m|d) + \frac{(1-\alpha) \cdot \sum_{n=1}^{m-1} \beta \cdot P(q_n|d) + (1-\beta) \cdot P(f_n|d)}{|I|}. \quad (3)$$

When $\beta = 1$, only the results from the user-issued query are taken into account, and conversely, when $\beta = 0$, only results from the feedback query are taken into account. We determine α and β by optimizing on the training data as described in Section 3.3.

2.2 Semantic Query Modeling

Wikipedia provides a rich and extensive source of information, not only in terms of content but also in terms of more structural information, such as the hyperlinks between articles. We implemented a novel algorithm that leverages the anchor texts of incoming hyperlinks to Wikipedia articles, including not only "normal" hyperlinks, but also redirects and alternative titles of pages. It does so in two steps. First, the anchor texts are used to identify and score relevant Wikipedia articles for all possible term n-grams in a query [7, 8]. Then, for each of these articles, we again use the incoming anchor texts. In this step, however, we use them to determine the parameters of a language model for each article. The language models are subsequently combined for all n-grams in the query, yielding as end result a semantically-informed language model of the query, i.e., a semantic query model (SQM).

2.3 Learning from Implicit Feedback

Our implicit feedback strategy uses machine learning to combine different ranking features, building on previous work in the area of online learning to rank from implicit feedback [3, 4]. We experiment with two learning strategies: (1) learning from explicit relevance judgments on an external collection, and (2) learning from previous result clicks on previous results in the same session and other sessions. Our experiments compare these two strategies to address the following research questions:

- Do weights tuned on an external collections carry over to the TREC session task?
- Do the clicks included with the TREC session data provide enough information for effective learning to rank?
- How do weights tuned using learning to rank from implicit feedback compare to those tuned on an external collection?

For both learning strategies we use the same implementation of an online learning to rank algorithm. Training data is constructed from either the external collection or click data observed on previous user interactions. The data is then used to learn weights for linear weighted combinations of the ranking features. To generate the final result list to be presented to the user, we apply the learned weight vector to the current query. Below, we detail the methods used to construct training data, the learning algorithm, features, and research questions.

Constructing training data In runs using an *external collection*, we constructed training instances from the data provided for the TREC 2009 Million Query track [1]. We used relevance assessments from the last queries of training sessions we constructed, see Section 3.3. In preliminary experiments, we found that best results were obtained when using all pairs of documents where one document was judged somewhat relevant (relevance grade 1), and the other document was judged highly relevant (grade 2). We extracted all such document pairs and trained our learning algorithm on all resulting data. As the algorithm is stochastic, we ran it repeatedly and selected the best weight vector to generate the results.

For the run using click data, we extract pair-wise preference relations, following [5]. In this approach, result documents that were clicked and presented at a lower rank than other, non-clicked documents, can be inferred to be more relevant than these non-clicked documents. Here, we extract all pairs that can be constructed in this way from all sessions.

Thus, given a document result list that was presented to a user in response to a query, and for which one or more clicks on documents were observed, we construct all possible ordered document pairs $\mathbf{X}$, and infer labels $Y(x) \in \{-1, +1\}$ for those pairs for which one document was clicked while the other was not (pairs for which neither, or both documents were clicked are discarded).

During learning, we use two sets of training data. *Global* training instances $\mathbf{X_G}$ include all pair-wise preference relations inferred for all but the current target session (similar to leave-one-out). *Local* training instances $\mathbf{X_L}$ include all training instances extracted for the current session (if available).

Features To facilitate learning that can generalize over documents and queries, we use a feature $\phi(d|q)$ representation that encodes the relationship between queries q and documents d. To apply pair-wise learning to rank we then represent pairs of documents by the difference of their feature vectors, such that $x_i = \phi(d_{i_1}|q) - \phi(d_{i_2}|q)$. As the amount of training data is fairly limited, we limited ourselves to the following three features:

- standard retrieval scores for the current query on the anchor text,
- scores returned by the run *UvAmodeling.RL[1-3]*, and
- scores returned by the run *UvAsemantic.RL[1-3]*.

Learning to rank Given a set of observed data $(\mathbf{X}, \mathbf{Y})$ for P document pairs, we apply the stochastic gradient descent (SGD) algorithm defined by Zhang [11, Algorithm 2.1]. This algorithm finds a weight vector $\hat{\mathbf{w}}$ that minimizes the empirical loss:

$$\hat{\mathbf{w}} = arg\min_{\mathbf{w}} \left[\frac{1}{P} \sum_{i=0}^{P} L(\mathbf{w}, \mathbf{x}_i, y_i) + \frac{\lambda}{2} ||\mathbf{w}||_2^2 \right], \qquad (4)$$

where $L(\mathbf{w},\mathbf{x},y)$ is a loss function, in this case the hinge loss, and the last term is a regularization term. The algorithm was shown to perform competitvely on standard learning to rank datasets [9]. Here, we follow the implementation provided in sofia-ml.[1] For each observed training sample $(\mathbf{x}_t, y_t)$, this algorithm updates the weight vector $\mathbf{w}_t$ using the update rule $\mathbf{w}_{t+1} = \mathbf{w}_t + \eta y_t \mathbf{x}_t - \eta\lambda\mathbf{w}$. We use the unregularized version of this algorithm (by setting $\lambda = 0$) and use a small constant $\eta = 0.0005$.

Finally, a given weight vector $\mathbf{w}$ is applied directly to the extracted feature vectors for given document-query pairs to compute ranking scores $\mathbf{S} = \mathbf{w}\phi(D|q)$. Documents are sorted by this score to obtain a result ranking [5].

3 Experimental Setup

3.1 Retrieval Framework

For retrieval we use a standard KL-divergence approach, with Bayesian smoothing using a Dirichlet prior.

3.2 Collection

We made use of the English portion of the Clueweb A document collection. We did not use any form of stemming and removed a conservative list of 588 stopwords. We removed the 70% of the documents most likely to be spam according to the Waterloo fused spam scores [2]. Some Wikipedia pages were classified as spam using this algorithm, however, we kept these pages in the collection as we consider Wikipedia to be valuable knowledge resource.

We created two indexes. For the first index, the *content* index, we stripped all html from the documents and indexed the resulting plain text using the Indri retrieval engine. For the second index, the *anchor-text* index, we represented each document by the anchor text of URLs pointing to that document.

3.3 Training Data

To create training sessions, we worked backwards from queries for which we had relevance assessments. We selected queries from the Million Query Track 2009. For each query, we wrote an information need for which the query would be a reasonable last query q_m. Then we performed search sessions in Clueweb09[2] based on that information need and we recorded the result pages and our clicks.

3.4 Runs

All submitted runs were Category A runs.

[1] Provided online at `http://code.google.com/p/sofia-ml/`.

[2] We used `http://boston.lti.cs.cmu.edu/Services/`

UvAmodeling Run based on explicit modeling. All runs are created according to Equation 3. Weight parameters are determined using the training collection.

- **RL1** Use current query only: $\alpha = 1$
- **RL2** Use current query and previous queries, but no result snippets: $\alpha = 0.6, \beta = 1$
- **RL3** Use current query, previous queries, and previous displayed results: $\alpha = 0.6, \beta = 0.9$
- **RL4** Use current query, previous queries, and previous clicked results: $\alpha = 0.4, \beta = 0.1$

UvAsemantic Run based on semantic query modeling (SQM). All runs are created using SQM, in combination with Equation 3. Weight parameters are determined using the training collection.

- **RL1** Use SQM for current query only: $\alpha = 1$
- **RL2** Use SQM for current query and previous queries, but no result snippets: $\alpha = 0.6, \beta = 1$
- **RL3** Use SQM for current query and previous queries, and create query from previously displayed results without SQM: $\alpha = 0.5, \beta = 0.5$
- **RL4** Use SQM for current query and previous queries, and create query from previously clicked results without SQM: $\alpha = 0.5, \beta = 0.8$

UvAlearning Runs combine several ranking features using learning to rank on an external collection (RL1-3) or from implicit feedback (RL4) as described in §2.3.

- **RL1** Learn weights using the external collection; features: *anchor text*, *UvAmodeling.RL1*, *UvAsemantic.RL1*
- **RL2** Learn weights using the external collection; features: *anchor text*, *UvAmodeling.RL2*, *UvAsemantic.RL2*
- **RL3** Learn weights using the external collection; features: *anchor text*, *UvAmodeling.RL3*, *UvAsemantic.RL3*
- **RL4** Uses the same features as *UvAlearning.RL3*, but learns weights using implicit feedback, as described in §2.3.

4 Results

Result overviews for our submissions are shown in Table 1 and Table 2. We show results according to nDCG@10, as this gives a good indication of the quality of the top 10 search results.

The best run type overall was UvAmodeling. For this run, we did not use query expansion or implicit learning, but linearly interpolated results from the current query with results from previous queries and result clicks. This approach

Table 1: Result overview for all runs in terms of nDCG@10, considering only those documents relevant to the subtopics of the last query as relevant. The highest score pfr each experimental condition is indicated in bold.

Run type	RL1	RL2	RL3	RL4
UvAmodeling	**0.238**	**0.238**	0.232	**0.254**
UvAsemantic	0.186	0.181	0.208	0.192
UvAlearning	0.211	0.199	**0.233**	0.173

Table 2: Result overview for all runs in terms of nDCG@10, considering those documents relevant to the whole topic of the session. The highest score for each experimental condition is indicated in bold.

Run	RL1	RL2	RL3	RL4
UvAmodeling	**0.337**	**0.337**	0.349	**0.412**
UvAsemantic	0.285	0.305	0.355	0.329
UvAlearning	0.321	0.324	**0.400**	0.293

gained the highest performance under experimental conditions RL1, RL2, and RL4. It also achieved the highest overall score. The highest performing run type under condition RL3 was UvAlearning.

Looking in more detail at the UvAmodeling run set, we see that performance did not increase between RL1 and RL2; this was due to a a bug in the system. This means that we cannot conclusively state whether incorporating results from previously issued queries improves performance for the current query in this setting. However, when we add information from previously displayed results (RL3), we see that performance goes down when evaluating with only the documents relevant to only the last subtopic. However, when evaluating with documents relevant to the whole topic of the session, performance increases. Finally, performance increases for both types of evaluation when we add information from previously clicked result snippets under RL4, and indeed this is where we attain the best performance.

Regarding our learning approaches, where weights for linear weighted combinations of runs were learned from either external data, or implicit feedback, we find few improvements over individual runs. The runs *UvAlearning.RL[1-2]* outperform the *UvAsemantic* runs, but score lower than the *UvAmodeling* runs. The reason may be that semantic modeling performs better on the external collection than on the TREC session track data, so that weights for this run are over estimated by the learning approach. In the *RL3* series, where scores for *UvAsemantic* are higher, the benefit of combining weights increases, so that *UvAlearning.RL3* outperforms all other runs in this series.

Moving to the run *UvAlearning.RL4*, we see that scores are substantially lower than for *UvAlearning.RL3*, which uses the same features and learning algorithm, but training data extracted from implicit feedback. We conclude that the click data associated with the current sessions does not contain enough information to provide useful feedback for learning to rank.

5 Conclusion

We have described the participation of the University of Amsterdam's ILPS group in the session track at TREC 2011. In our experiments we examined three complementary strategies for improving retrieval for a current query. Our first strategy, based on probabilistic session modeling, was the best performing strategy.

Our second strategy, based on semantic query modeling, did less well than we expected, likely due to topic drift from excessively aggressive query expansion. We expect that performance of this strategy would improve by limiting the number of terms and/or improving the probability estimates.

With respect to our third strategy, based on learning from feedback, we found that learning weights for linear weighted combinations of features from an external collection can be beneficial, if characteristics of the collection are similar to the current data. Feedback available in the form of user clicks appeared to be less beneficial. Our run learning from implicit feedback did perform substantially lower than a run where weights were learned from an external collection with explicit feedback using the same learning algorithm and set of features.

Acknowledgments

This research was partially supported by the European Union's ICT Policy Support Programme as part of the Competitiveness and Innovation Framework Programme, CIP ICT-PSP under grant agreement nr 250430, the PROMISE Network of Excellence co-funded by the 7th Framework Programme of the European Commission under grant agreement nr 258191, the LiMoSINe project co-funded by the 7th Framework Programme of the European Commission under grant agreement nr 288024, the Netherlands Organisation for Scientific Research (NWO) under project nrs 612.061.-814, 612.061.815, 640.004.802, 380-70-011, 727.011.005, the Center for Creation, Content and Technology (CCCT), the Hyperlocal Service Platform project funded by the Service Innovation & ICT program, the WAHSP project funded by the CLARIN-nl program, under COMMIT project Infiniti and by the ESF Research Network Program ELIAS.

6 References

[1] Carterette, B., Pavlu, V., Fang, H., and Kanoulas, E. (2010). Million query track 2009 overview. In *The Eighteenth Text Retrieval Conference Proceedings (TREC 2009)*. NIST. Special Publication.

[2] Cormack, G. V., Smucker, M. D., and Clarke, C. L. A. (2010). Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval*, pages 1–25.

[3] Hofmann, K., Whiteson, S., and de Rijke, M. (2011a). Balancing exploration and exploitation in learning to rank online. In *ECIR 2011: 33rd European Conference on Information Retrieval*.

[4] Hofmann, K., Whiteson, S., and de Rijke, M. (2011b). A probabilistic method for inferring preferences from clicks. In *20th ACM Conference on Information and Knowledge Management (CIKM 2011)*.

[5] Joachims, T. (2002). Optimizing search engines using clickthrough data. In *KDD '02*, pages 133–142.

[6] Lavrenko, V. and Croft, W. B. (2001). Relevance based language models. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*.

[7] Meij, E. and de Rijke, M. (2010). Supervised query modeling using Wikipedia. In *SIGIR '10: Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*.

[8] Meij, E., Weerkamp, W., and de Rijke, M. (2012). Adding semantics to microblog posts. In *Proceedings of the fifth ACM international conference on Web search and data mining (WSDM 2012)*.

[9] Sculley, D. (2009). Large scale learning to rank. In *NIPS 2009 Workshop on Advances in Ranking*.

[10] Shen, X., Tan, B., and Zhai, C. (2005). Context-sensitive information retrieval using implicit feedback. In *SIGIR*, pages 43–50. ACM.

[11] Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML '04*, pages 116+. ACM.